

Computer algebra compares the stochastic superslow manifold of an averaged SPDE with that of the original slow-fast SPDE

A. J. Roberts*

March 22, 2011

Abstract

The computer algebra routines¹ documented here empower you to reproduce and check many of the details described by an article on large deviations for slow-fast stochastic systems [6]. We consider a ‘small’ spatial domain with two coupled concentration fields, one governed by a ‘slow’ reaction-diffusion equation and one governed by a stochastic ‘fast’ linear equation. In the regime of a stochastic bifurcation, we derive two superslow models of the dynamics: the first is of the averaged model of the slow dynamics derived via large deviation principles; and the second is of the original fast-slow dynamics. Comparing the two superslow models validates the averaging in the large deviation principle in this parameter regime [6].

Contents

1	Iterative computer algebra derives the model	2
1.1	Initialisation	3

*School of Mathematical Sciences, University of Adelaide, South Australia. <http://www.maths.adelaide.edu.au/anthony.roberts>

¹The computer algebra routines are written in the free package REDUCE. At the time of writing, <http://www.reduce-algebra.com/> provides information about REDUCE.

<i>1 Iterative computer algebra derives the model</i>	<i>2</i>
1.2 Compute residual	5
1.3 Update the stochastic slow manifold	6
1.4 Linear noise effects	6
1.5 Quadratic noise effects	9
1.6 Transform quadratic noise	11
2 Model interacting fast-slow-superslow components	13
2.1 Some initialisation things	16
2.2 Update from residuals of the fast equation	24
2.3 Update from residuals of the slow equation	24
References	25

1 Iterative computer algebra derives the model

Construct a one element model of the ‘slow’ stochastic reaction diffusion equation

u_t = d_{xx}u + λ sin u - (1 - d_{xx})^{-1}u - √εσ(1 - d_{xx})^{-1}ϕ(x,t) (1)

such that u = 0 at x = 0, π,

near the deterministic bifurcation that occurs at λ = 3/2, to effects quadratic in the noise amplitude σ. We seek the normal form where the evolution involves no convolutions [1, 5]. Also, transform the quadratic noise in the evolution. Throughout we adopt the Stratonovich interpretation of stochastic differential equations so that the ordinary rules of calculus apply.

The stochastic slow model appears to be, when parameter λ = 3/2 + λ' and upon truncating the noise to just the first three sine modes,

a˙ = λ'a - (3/16 + 1/8λ')a^3 + 91/9728a^5 - √εσ(1/2ϕ_1 + 3/1216a^2ϕ_3) + ... (2)

when the stochastic slow manifold is

u = a sin x + 5/608a^3 sin 3x

$$\begin{aligned}
& -\sqrt{\epsilon}\sigma \left\{ \frac{1}{5} \sin 2x e^{-\frac{27}{10}t} \star \phi_2 + \frac{1}{10} \sin 3x e^{-\frac{38}{5}t} \star \phi_3 \right. \\
& \left. + \lambda' \left[\frac{1}{5} \sin 2x e^{-\frac{27}{10}t} \star e^{-\frac{27}{10}t} \star \phi_2 + \frac{1}{10} \sin 3x e^{-\frac{38}{5}t} \star e^{-\frac{38}{5}t} \star \phi_3 \right] \right\} + \dots
\end{aligned}$$

In outline, the algorithm iteratively determines the stochastic slow manifold model, then finally transforms to a weak model by replacing quadratic noises by their long time equivalents. Earlier research [2] explained the centre manifold rationale and the computational effectiveness of this simple algorithm, albeit there restricted to deterministic systems.

```

>> ssmaveq <<

% see cassmaveq.pdf for documentation
<< initialisation >>
<< linear noise effects >>
<< quadratic noise effects >>
sig:=small*sigma;
let { small^6=>0 };
it:=1$
repeat begin
<< compute residual >>
<< update ssm >>
  showtime;
end until res=0 or (it:=it+1)>20;
write gssm:=sub(small=1,g);
<< transform quadratic noise >>
end;

```

1.1 Initialisation

Trivially improve printing.

```

>> initialisation <<
on div; off allfac; on revpri;

```

```
factor sigma,sin,small;
linelength 65$
```

Define the parameter λ to be a small perturbation away from critical. Scale this with ordering parameter **small** in order to control truncation in the multiple small parameters.

```
>> initialisation <<+
lamb:=3/2+small^2*lam;
```

Linearise products of trigonometric functions via **trigsimp**.

Define α_m to be the decay rate of linear modes, here $\alpha_m = m^2 - 3/2 + 1/(m^2 + 1)$, so that the spatial modes decay linearly like $\sin(mx)e^{-\alpha_m t}$.

```
>> initialisation <<+
procedure alfa(m); (m^2-3/2+1/(m^2+1))$
```

Define the inverse of the linear operator, $\mathcal{L}^{-1} \sin(mx) = \sin(mx)/\alpha_m$, as the linear operator is $\mathcal{L} = -3/2 - \partial_{xx} + (1 - \partial_{xx})^{-1}$. Note: we only define and use this for $m \geq 2$.

```
>> initialisation <<+
operator uinv; linear uinv;
let uinv(sin(~m*x),xt)=>sin(m*x)/alfa(m);
```

Define the linear operator, $(1 - \partial_{xx})^{-1} \sin(nx)$.

```
>> initialisation <<+
operator iddi; linear iddi;
let { iddi(sin(~n*x),x) => sin(n*x)/(1+n^2)
    , iddi(sin(x),x) => sin(x)/2 };
```

Paramterise solutions by an evolving amplitude $\bar{a}(t)$ (or ‘order parameter’). Its evolution is $d\bar{a}/dt = \dot{\bar{a}} = g$.

```

>> initialisation <<+
depend a,t;
let df(a,t)=>g;

```

Then the most basic linear approximation to the dynamics on the element is $\bar{u} = \bar{a} \sin x$ where $\dot{\bar{a}} = 0$. Scale the amplitude to be small.

```

>> initialisation <<+
u:=small*a*sin(x);
v:=u/2;
g:=0;

```

1.2 Compute residual

The parameter **small** conveniently controls the truncation in nonlinearity and other small parameters. The iteration terminates when the residual of the reaction diffusion equation is zero to the specified order of smallness. Note: this parametrisation with **small** should create deterministic models to errors $\mathcal{O}(\bar{a}^4 + \epsilon^2)$, or equivalent, as we scale $\epsilon = \mathbf{eps}$ with **small**². For some strange reason we need to do some operation on **res** in order for relevant terms to cancel—here I use **trigsimp**, but something else might serve.

```

>> compute residual <<
sinu:=trigsimp(u-u^3/6+u^5/120-u^7/5040,combine);
res:=-df(u,t)+df(u,x,2)+lamb*sinu-iddi(u,x)
      -small*rooteps*sig*iddi(noise,x);
res:=trigsimp(res,combine);
write lengthres:=length(res);

```

Define $\sqrt{\epsilon}$ which scales the size of the noise. Looks like we do not have to worry about `rooteps/eps` not simplifying.

```
>> initialisation <<+
let rooteps^2=>eps;
```

1.3 Update the stochastic slow manifold

Let T , `tt`, label the fast time of stochastic fluctuations so we can separate the stochastic fluctuations from the superslow evolution of the amplitude \bar{a} . Also introduce `xt` to label both the subgrid spatial scales and time scales so we can group all factors in the space-time dynamics.

```
>> initialisation <<+
depend tt,t;
depend x,xt;
```

Then update driven by the residual. Divide the evolution `g` by `small` to best keep track of the correct counting of the ‘order’ of a term.

```
>> update ssm <<
g:=g+(gd:=secular(res,xt))/small;
u:=u+uinv(res-gd*sin(x),xt);
```

1.4 Linear noise effects

Introduce the noise in its spatial Fourier decomposition

$$\phi(x, t) = \sum_{n=1}^{\infty} \phi_n(t) \sin nx.$$

Parametrise the amplitude with σ . Truncate the spatial structure of noise. Three terms in the noise appears adequate to show the typical interactions between noise and other dynamics. Have not explored details of better resolution of the noise.

```

>> linear noise effects <<
operator phi; depend phi,tt,xt;
noise:=for n:=1:3 sum phi(n,{})*sin(n*x);

```

Let $\text{phi}(n,\{m_1,\dots\})$ denote convolutions with $\exp(-m_1 t) \dots$, that is,

$$\phi_{n,(m_1,m_2,\dots)} = \exp(-m_1 t) \star \exp(-m_2 t) \star \dots \star \phi_n(t);$$

so

$$\partial_t \phi_{n,(m_1,m_2,\dots)} = -m_1 \phi_{n,(m_1,m_2,\dots)} + \phi_{n,(m_2,\dots)}.$$

But if we pull out a decay rate in $1/\epsilon$ then keep bookkeeping correct by dividing by `small^2` unless it is a *sole* convolution by $\mathcal{O}(1/\epsilon)$. This latter case is only used in the next section.

```

>> linear noise effects <<+
let { df(phi(~m,~p),t)=>df(phi(~m,~p),tt)
    , df(phi(~m,~p),tt)=>(-first(p)*phi(m,p)+phi(m,rest(p)))
      when deg(1/first(p),eps)=0
    , df(phi(~m,~p),tt)=>(-first(p)*phi(m,p)/small^2
      +phi(m,rest(p))/small^(if rest(p)={} then 1 else 2))
      when deg(1/first(p),eps)=1
    };

```

Recall the equation for updates \bar{u}' and g' is $g' + \mathcal{L}\bar{u}' = \text{residual}$, where now the operator $\mathcal{L} = \partial_t - 3/2 - \partial_{xx} + (1 - \partial_{xx})^{-1}$ includes fast time variations. The operator `secular` extracts from the residual all those terms which would generate secular growth in the field u and so instead must be placed in the model's evolution g . The last rule here comes from integration by

parts and is essential in order to eliminate memory integrals (convolutions) in the model evolution.

The if-clause in the last is needed for the next section to account for the convolution on the fast time only being of $\mathcal{O}(\sqrt{\epsilon})$, and so order increases

```

>> linear noise effects <<+
operator secular; linear secular;
let { secular(sin(~m*x),xt)=>0
      , secular(sin(~m*x)*~aa,xt)=>0
      , secular(sin(x),xt)=>1
      , secular(sin(x)*phi(~n,~p),xt)=>
        phi(n,{})*(for each r in p product (1/r))
        *(if p neq{} and deg(1/first(p),eps)=1 then small else 1)
      };

```

Extend the inverse operator to terms with fast time variations as well as fast (subgrid) space variations. Recursive procedure **gungb** extracts the non-secular parts of fluctuating sin x components. Have to adjust the smallness whenever the convolution transformed is on the ϵ scale.

```

>> linear noise effects <<+
procedure gungb(n,p);
  if p={} then 0 else
    (gungb(n,rest(p))-phi(n,p)
     *(if deg(1/first(p),eps)=0 then 1 else small^2)
     )/first(p)$

let { uinv(sin(~m*x)*phi(~n,~p),xt)=>phi(n,(alfa(m)).p)*sin(m*x)
      , uinv(sin(x)*phi(~n,~p),xt)=>gungb(n,p)*sin(x)
      };

```


1.5 Quadratic noise effects

Now let Z_p denote multiple convolutions of in time of any term, $zz(a,p)$ (though I only use Z for quadratic terms, it may well be able to replace the linear convolutions). That is,

$$Z_{(m_1, m_2, \dots)} = \exp(-m_1 t) \star Z_{(m_2, \dots)} \quad \text{and} \quad Z_{()} = 1.$$

```

    >> quadratic noise effects <<
operator zz; depend zz,tt,xt;
let { zz(~a,{})=>a
    , df(zz(~a,~p),t)=>df(zz(a,p),tt)
    , df(zz(~a,~p),tt)=>-first(p)*zz(a,p)+zz(a,rest(p))
      when deg(1/first(p),eps)=0
    };

```

To extract quadratic corrections to the evolution, use integration by parts so all non-integrable convolutions are reduced to the canonical form of the convolution being entirely over one noise in a quadratic term, either $\phi_n \phi_{m,(\dots)}$ or $\phi_{n,(\dots)} \phi_m$.

Have now made this very complicated for at least some of the cases when the convolutions may be over ϵ -fast time scales.

```

    >> quadratic noise effects <<+
procedure gungd(n,p,m,q);
  if (p={})or(q={}) then phi(n,p)*phi(m,q)
  else if deg(1/first(p),eps)=deg(1/first(q),eps) then
    (gungd(n,rest(p),m,q)+gungd(n,p,m,rest(q)))
    /(first(p)+first(q))
  *(if deg(1/first(p),eps)=0 then 1 else small^2)
  else if deg(1/first(p),eps)=1 then
    (gungd(n,rest(p),m,q)*(if rest(p)={} then small else small
      +gungd(n,p,m,rest(q))*small^2
    )/first(p)*sub(rat=-first(q)/first(p),geom)

```

```

else gungd(m,q,n,p)$

let { secular(sin(x)*zz(~a,~p),xt) =>secular(sin(x)*a,xt)
      *(for each r in p product (1/r))
      , secular(sin(~m*x)*zz(~a,~p),xt)=>0
      , secular(sin(x)*phi(~n,~p)*phi(~m,~q),xt) =>gungd(n,p,m,q)
      , secular(sin(x)*phi(~n,~p)^2,xt) =>gungd(n,p,n,p)
    };

```

Extend \mathcal{L}^{-1} operator `uinv` to handle quadratic terms. First, integration by parts gives all integrable contributions from direct product terms.

Have to similarly modify `gunge` for at least some of the cases when the convolutions may be over ϵ -fast time scales.

```

    >> quadratic noise effects <<<+
procedure gunge(n,p,m,q);
  if (p={})or(q={}) then 0
  else if deg(1/first(p),eps)=deg(1/first(q),eps) then
    (-phi(n,p)*phi(m,q)
      *(if deg(1/first(p),eps)=0 then 1 else small)^2
      +gunge(n,rest(p),m,q)
      +gunge(n,p,m,rest(q))
    )/(first(p)+first(q))
  else if deg(1/first(p),eps)=1 then
    (-phi(n,p)*phi(m,q)*small^2
      +gunge(n,rest(p),m,q)*(if rest(p)={} then small else sma
      +gunge(n,p,m,rest(q))*small^2
    )/first(p)*sub(rat=-first(q)/first(p),geom)
  else gunge(m,q,n,p)$
let { uinv(sin(x)*phi(~n,~p)*phi(~m,~q),xt)
      =>gunge(n,p,m,q)*sin(x)
      , uinv(sin(x)*phi(~n,~p)^2,xt)=>gunge(n,p,n,p)*sin(x)
    };

```

Second, similar integration by parts gives integrable contribution from terms involving convolutions of products.

```

    >> quadratic noise effects <<+
procedure gungf(a,p);
  if p={} then 0 else
    (gungf(a,rest(p))-zz(a,p))/first(p)$
let { uinv(sin(~l*x)*phi(~n,~p)*phi(~m,~q),xt)
      =>sin(l*x)*zz(phi(n,p)*phi(m,q),{alfa(l)})}
, uinv(sin(~l*x)*phi(~n,~p)^2,xt)
      =>sin(l*x)*zz(phi(n,p)^2,{alfa(l)})}
, uinv(sin(~l*x)*zz(~a,~p),xt)=>sin(l*x)*zz(a,alfa(l).p)
, uinv(sin(x)*zz(~a,~p),xt)=>sin(x)*gungf(a,p)
};

```

1.6 Transform quadratic noise

Now proceed to transform the strong model to a weak model by replacing the quadratic noises by their effective long term drift and volatility. Earlier work [3] determines the rationale for the details of this transformation.

Set `small = 1` as it has done its job of truncating the nonlinear terms in the asymptotic expansion.

```

    >> transform quadratic noise <<
small:=1;
write "Now transforming the quadratic noises";

```

Now transform the quadratic noise into new noises ψ (**psi**) that are equivalent in their long time statistics: the operator **long** implements the long-time equivalent noises as determined earlier [3]. For now only transform up to two convolutions. These new noises have subscripts that uniquely identify them.

```

    >> transform quadratic noise <<+
operator long; linear long;
operator psi; depend psi,tt,xt;
let { long(1,tt)=>1
    , long(phi(~i,{}),tt)=>phi(i,{})
    , long(phi(~i,{})*phi(~j,{~k}),tt)
    => 1/2*(if i=j then 1 else 0)
    +psi(i,j,{k})/sqrt(2*k)
    , long(phi(~i,{})*phi(~j,{~k2,~k1}),tt)
    => (psi(i,j,{k1})/sqrt(2*k1)
    +psi(i,j,{k2,k1})/sqrt(2*k2))/(k1+k2)
    };
gg:=long(g,tt)$

```

Root sum squares of the determined noise coefficients; this procedure implicitly assumes that there is no correlation between the multitude of noises in these two terms in the amplitude equation. This assumption appears correct for these two terms in this PDE. In general we would need to do a QR factorisation of the noise terms.

```

    >> transform quadratic noise <<+
operator sumsqpsi; linear sumsqpsi;
let { sumsqpsi(1,tt)=>0
    , sumsqpsi(psi(~i,~j,~p),tt)=>0
    , sumsqpsi(psi(~i,~j,~p)^2,tt)=>1
    , sumsqpsi(psi(~i,~j,~p)*psi(~ii,~jj,~pp),tt)=>0
    };

```

Have a look at the numerical coefficients.

```

    >> transform quadratic noise <<+
on rounded; print_precision 5;
gg:=gg;

```

Extract the coefficients of the terms in σ^2 and $\sigma^2 \mathbf{a}$, both mean and fluctuating.

```

>> transform quadratic noise <<+
let abs(eps)=>eps;
c20:=sqrt(sumsqpsi(coeffn(coeffn(gg,sig,2),a,0)^2,tt));
c21mean:=(coeffn(coeffn(gg,sig,2),a,1)
  where psi(~i,~j,~p)=>0);
c21:=sqrt(sumsqpsi(coeffn(coeffn(gg,sig,2),a,1)^2,tt));

```

Switch back to the rational arithmetic mode for any other postprocessing.

```

>> transform quadratic noise <<+
off rounded;
showtime;

```

Executing the resultant code constructs the superslow model of the stochastic bifurcation in the slow averaged SPDEs.

2 Model interacting fast-slow-superslow components

This section constructs a superslow model of the fast-slow stochastic reaction diffusion equation

$$\mathbf{u}_t = \partial_{xx} \mathbf{u} + \lambda \sin \mathbf{u} - \mathbf{v}, \quad (3)$$

$$\epsilon \mathbf{v}_t = \partial_{xx} \mathbf{v} - \mathbf{v} + \mathbf{u} + \sqrt{\epsilon} \sigma \phi(x, t), \quad (4)$$

$$\text{such that } \mathbf{u} = \mathbf{v} = 0 \text{ at } x = 0, \pi,$$

near the deterministic bifurcation that occurs at $\lambda = 3/2$, to effects quadratic in the noise amplitude σ , and seeks the normal form where the evolution involves no convolutions. Recall that throughout we adopt the Stratonovich interpretation of stochastic differential equations so that the ordinary rules of calculus apply.

The presence of the three time scales in the dynamics—the fast \mathbf{v} , the slow \mathbf{u} , and the superslow evolution of the bifurcation amplitude \mathbf{a} —means that the algorithm outlined here is one of the most technically challenging ones I have implemented. I recommend understanding simpler systems before attempting to understand the details of this section.

The resulting model appears to be the following to some order in small parameters. In terms of the superslow evolving amplitude $\mathbf{a}(\mathbf{t})$, where $\mathbf{u} \approx \mathbf{a} \sin \mathbf{x}$ and $\mathbf{v} \approx \frac{1}{2} \mathbf{a} \sin \mathbf{x}$, parameter $\lambda = \frac{3}{2} + \lambda'$, and noise in just three sine modes, a stochastic differential equation for the amplitude is

$$\begin{aligned} \dot{\mathbf{a}} = & \lambda'(1 + \tfrac{1}{4}\epsilon\lambda')\mathbf{a} - (\tfrac{3}{16} + \tfrac{1}{8}\lambda' + \tfrac{3}{64}\epsilon)\mathbf{a}^3 + \tfrac{91}{9728}\mathbf{a}^5 \\ & - \sqrt{\epsilon}\sigma(\tfrac{1}{2} + \tfrac{1}{8}\epsilon - \tfrac{1}{4}\epsilon\lambda' + \tfrac{9}{64}\epsilon\mathbf{a}^2)\phi_1 \\ & - \sqrt{\epsilon}\sigma(\tfrac{3}{1216} + \tfrac{3}{4864}\epsilon)\mathbf{a}^2\phi_3 \\ & + \epsilon\sigma^2\mathbf{a} \left[-\tfrac{1}{180}\phi_2 e^{-\frac{27}{10}\mathbf{t}} \star \phi_2 + \tfrac{3}{1216}\phi_1 e^{-\frac{38}{5}\mathbf{t}} \star \phi_3 - \tfrac{6}{6080}\phi_3 e^{-\frac{38}{5}\mathbf{t}} \star \phi_3 \right] \\ & + \dots \end{aligned} \tag{5}$$

To errors $\mathcal{O}(\epsilon)$, this evolution equation is identical to the slow model (2) of the averaged equation with fluctuations.

The corresponding stochastic superslow manifold appears to be that the slow field

$$\begin{aligned} \mathbf{u} = & \mathbf{a} \sin \mathbf{x} + \tfrac{5}{608}\mathbf{a}^3 \sin 3\mathbf{x} + \tfrac{1}{2}\sqrt{\epsilon}\sigma \sin \mathbf{x} e^{-\frac{2}{\epsilon}\mathbf{t}} \star \phi_1 \\ & - \tfrac{1}{5}\sqrt{\epsilon}\sigma \sin 2\mathbf{x} \left[e^{-\frac{27}{10}\mathbf{t}} \star -e^{-\frac{5}{\epsilon}\mathbf{t}} \star \right] \phi_2 \\ & - \tfrac{1}{10}\sqrt{\epsilon}\sigma \sin 3\mathbf{x} \left[e^{-\frac{38}{5}\mathbf{t}} \star -e^{-\frac{10}{\epsilon}\mathbf{t}} \star \right] \phi_3 + \dots \end{aligned}$$

whereas the fast field has $\mathcal{O}(1)$ fluctuations

$$\begin{aligned} \mathbf{v} = & \tfrac{1}{2}\mathbf{a} \sin \mathbf{x} + \tfrac{1}{1216}\mathbf{a}^3 \sin 3\mathbf{x} \\ & + \tfrac{\sigma}{\sqrt{\epsilon}} \sin \mathbf{x} \left[(1 + \tfrac{1}{4}\epsilon) e^{-\frac{2}{\epsilon}\mathbf{t}} \star + \tfrac{1}{2} e^{-\frac{2}{\epsilon}\mathbf{t}} \star e^{-\frac{2}{\epsilon}\mathbf{t}} \star \right] \phi_1 \end{aligned}$$

$$\begin{aligned}
& + \frac{\sigma}{\sqrt{\epsilon}} \sin 2x \left[\left(1 + \frac{1}{25}\epsilon\right) e^{-\frac{5}{\epsilon}t} \star -\epsilon \frac{1}{25} e^{-\frac{27}{10}t} \star + \frac{1}{5} e^{-\frac{5}{\epsilon}t} \star e^{-\frac{5}{\epsilon}t} \star \right] \phi_2 \\
& + \frac{\sigma}{\sqrt{\epsilon}} \sin 3x \left[\left(1 + \frac{1}{100}\epsilon\right) e^{-\frac{10}{\epsilon}t} \star -\epsilon \frac{1}{100} e^{-\frac{38}{5}t} \star + \frac{1}{10} e^{-\frac{10}{\epsilon}t} \star e^{-\frac{10}{\epsilon}t} \star \right] \phi_3 \\
& + \dots
\end{aligned}$$

In outline, the algorithm iteratively determines the stochastic superslow manifold model [2, e.g.].

Seem to have to keep one or two orders higher in `small` as there is a division somewhere. So actually compute to residuals and errors one or two orders in `small` less than apparently allowed here—this fudges the computations so that enough is kept to do the cancellation, then later truncates. That is, `small^6=>0` actually computes to errors $\mathcal{O}(\text{small}^4)$. Here, for some reason we have to tread carefully to get up to fifth order terms in `small`: first, compute linear noise effects; then, second, when residuals are zero, up the order to retain quadratic terms in noise and continue iterating. Alternatively, we could just seek terms up to fourth order in `small` by `let small^7=>0`. But I do want to get to fifth order because of the challenge.

```

>> ssmuv <<

% see cassmaveq.pdf for documentation
let { sigma^2=>0, small^8=>0 };
sig:=small*sigma;
<< initialisation >>
<< linear noise effects >>
<< quadratic noise effects >>
it:=1$
repeat begin
  << update from fast residual >>
  if {resu,resv}={0,0} then clear sigma^2;%implicitly sigma^3=>0
  << update from slow residual >>
    showtime;
end until {resu,resv}={0,0} and (sigma^2neq0) or (it:=it+1)>19
%write ussm:=sub(small=1,u);
%write vssm:=sub(small=1,v);

```

Table 1: order of magnitude of convolution operators.

	1	$e^{-\alpha t_\star}$	$(e^{-\alpha t_\star})^2$	$(e^{-\alpha t_\star})^3$
1	1	1	1	1
$e^{-\frac{\beta}{\epsilon} t_\star}$	$\mathcal{O}(\epsilon^{1/2})$	$\mathcal{O}(\epsilon)$	$\mathcal{O}(\epsilon)$	$\mathcal{O}(\epsilon)$
$(e^{-\frac{\beta}{\epsilon} t_\star})^2$	$\mathcal{O}(\epsilon^{3/2})$	$\mathcal{O}(\epsilon^2)$	$\mathcal{O}(\epsilon^2)$	$\mathcal{O}(\epsilon^2)$
$(e^{-\frac{\beta}{\epsilon} t_\star})^3$	$\mathcal{O}(\epsilon^{5/2})$	$\mathcal{O}(\epsilon^3)$	$\mathcal{O}(\epsilon^3)$	$\mathcal{O}(\epsilon^3)$

```

write gssm:=sub(small=1,g);
<< transform quadratic noise >>
end;

```

2.1 Some initialisation things

Define β_m to be the relative decay rate of linear modes of the fast variable v on the element, here $\beta_m = m^2 + 1$, so that the spatial modes in v decay linearly like $\sin(mx) \exp(-\beta_m t/\epsilon)$.

```

>> initialisation <<+
procedure beta(m); (m^2+1)$

```

Define some of the inverse of a linear operator. Now there are significant subtleties here: each convolution with rates $\mathcal{O}(1/\epsilon)$ are themselves. Thus smallness is hidden in the convolution rates; consequently we have to track them artificially though a parameter such as `small`. Use `small` to count both the direct ϵ and the hidden ones in the convolutions, as well as the other small parameters.

A further complication is that single bare convolution is actually $\mathcal{O}(\sqrt{\epsilon})$ [4, equation (27)]; Table 1 lists the correct order of magnitude of various convolutions. This complication is simplified a little by separating convolutions that occur on the fast time from the convolutions that occur on the low time scale.

The linear equations for updates are

$$\begin{aligned} -g \sin x - u_t + u_{xx} + \frac{3}{2}u - v + \text{Res}_u &= 0, \\ -\epsilon v_t + u + v_{xx} - v + \text{Res}_v &= 0. \end{aligned}$$

When considering mode $\sin mx$, the linear equations for updates are, including a correction g to the evolution only in the case of the critical $m = 1$,

$$\begin{aligned} -g - u_t - (\alpha_m - 1/\beta_m)u - v + \text{Res}_u &= 0, \\ -\epsilon v_t + u - \beta_m v + \text{Res}_v &= 0, \end{aligned}$$

for the previous defined constants α_m and β_m . The details of solving for updates are not critical to correctness of the results (as the results should only depend upon driving the residuals to zero), but the details will determine whether the iteration does converge to zero the residuals.

Updates from the u equation For residuals of the slow-equation, make updates to the u -field driven by the u -residual, and correspondingly update the v -field in a way that will not change its v -residual at this order. We insist on not changing the v -residual because this update is considered second and we must not undo earlier corrections driven from the v -residual. Dividing the u -update by β_m is sufficient for the v -update. List here first the deterministic updates, second the generic linear noise update, and last the updates for resonant terms. Use procedure **gungb** to extract the non-resonant parts of Res_u .

Account for smoothing effect of convolution on noise via the if-clauses. Table 1 shows that when a term goes from multiple fast time convolutions to include one additional slow time convolution, then the order of the term increases by $\sqrt{\epsilon}$ (the following provision assumes we linearise convolutions so that any one term only has convolutions on the same time scale).

```

>> linear noise effects <<+
operator uuin; linear uuin;
operator vuinv; linear vuinv;
let { uuinv(sin(~m*x),xt)=>sin(m*x)/alfa(m)

```

```

, vuinv(sin(~m*x),xt)=>sin(m*x)/alfa(m)/beta(m)
, uuinv(sin(~m*x)*phi(~n,~p),xt)
  => phi(n,(alfa(m)).p)*sin(m*x)
  *(if p neq{} and deg(1/first(p),eps)=1 then small else
, vuinv(sin(~m*x)*phi(~n,~p),xt)
  => phi(n,(alfa(m)).p)*sin(m*x)/beta(m)
  *(if p neq{} and deg(1/first(p),eps)=1 then small else
, uuinv(sin(x)*phi(~n,~p),xt)=>gungb(n,p)*sin(x)
, vuinv(sin(x)*phi(~n,~p),xt)=>gungb(n,p)*sin(x)/beta(1)
};

```

To deal with quadratic noise, the following appear to be enough for errors no higher order than `small`⁸. That is, with the modifications made to `gunge` and `gungd`. We do not seem to need any extra transformations from the residual of the fast v-equation, probably because it is linear.

```

>> quadratic noise effects <<+
let { uuinv(sin(~m*x)*phi(~n,~p)*phi(~l,~q),xt)
  => zz(phi(n,p)*phi(l,q),{alfa(m)})*sin(m*x)
  *(if p neq{} and deg(1/first(p),eps)=1 then small else
, vuinv(sin(~m*x)*phi(~n,~p)*phi(~l,~q),xt)
  => zz(phi(n,p)*phi(l,q),{alfa(m)})*sin(m*x)/beta(m)
  *(if p neq{} and deg(1/first(p),eps)=1 then small else
, uuinv(sin(~m*x)*zz(~n,~p),xt) => zz(n,alfa(m).p)*sin(m*x)
, vuinv(sin(~m*x)*zz(~n,~p),xt)
  => zz(n,alfa(m).p)*sin(m*x)/beta(m)
, uuinv(sin(~m*x)*phi(~n,~p)^2,xt)
  => zz(phi(n,p)^2,{alfa(m)})*sin(m*x)
  *(if p neq{} and deg(1/first(p),eps)=1 then small else
, vuinv(sin(~m*x)*phi(~n,~p)^2,xt)
  => zz(phi(n,p)^2,{alfa(m)})*sin(m*x)/beta(m)
  *(if p neq{} and deg(1/first(p),eps)=1 then small else
, uuinv(sin(x)*phi(~n,~p)*phi(~l,~q),xt)=>gunge(n,p,l,q)*s
, vuinv(sin(x)*phi(~n,~p)*phi(~l,~q),xt)
  => gunge(n,p,l,q)*sin(x)/beta(1)

```

```

, uuinv(sin(x)*phi(~n,~p)^2,xt)=>gunge(n,p,n,p)*sin(x)
, vuinv(sin(x)*phi(~n,~p)^2,xt)=>gunge(n,p,n,p)*sin(x)/beta(1)
, uuinv(sin(x)*zz(~n,~p),xt)=>
  ( uuinv(sin(x)*zz(n,rest(p)),xt)
    -zz(n,p)*sin(x) )/first(p)
, vuinv(sin(x)*zz(~n,~p),xt)=>
  ( vuinv(sin(x)*zz(n,rest(p)),xt)
    -zz(n,p)*sin(x) )/first(p)/beta(1)
};

```

Updates from the v equation Corrections to the u and v fields arise from the v -residual. However, because we consider this residual first in each iteration (not that first makes a lot of sense in an iterative loop), we are free to modify field u in a way that would affect the residuals at the same order. The key aspect is that we must not affect the residuals at a lower order in the u -residual—achieving this aspect is hard enough, which is why I implement corrections from the v -residual first.

First define the deterministic updates.

```

>> linear noise effects <<+
operator vvinv; linear vvinv;
operator uvinv; linear uvinv;
let { vvinv(sin(~m*x),xt)=>sin(m*x)*(alfa(m)-1/beta(m))
      /alfa(m)/beta(m)
    , uvinv(sin(~m*x),xt)=>-sin(m*x)/alfa(m)/beta(m)
    , vvinv(sin(x),xt)=>sin(x)/beta(1)
    , uvinv(sin(x),xt)=>0

```

Second deal with the variety of linear noise terms. When a noise term in the residual is a convolution over the slow-scale, then the convolution is smooth and its time derivative correspondingly of the same order so that the ev_t causes no problem.

```

>> linear noise effects <<+
, vvinv(sin(~m*x)*phi(~n,~p),xt)
  => phi(n,p)*sin(m*x)/beta(m)
  when p neq {} and deg(1/first(p),eps)=0
, uvinv(sin(~m*x)*phi(~n,~p),xt) => 0
  when p neq {} and deg(1/first(p),eps)=0

```

The critical mode is no different when forcing in the v -residual.

```

>> linear noise effects <<+
, vvinv(sin(x)*phi(~n,~p),xt)
  => phi(n,p)*sin(x)/beta(1)
  when p neq {} and deg(1/first(p),eps)=0
, uvinv(sin(x)*phi(~n,~p),xt) => 0
  when p neq {} and deg(1/first(p),eps)=0

```

However, when the noise term in the v -residual is not smooth, either because it is a bare white noise or because it is a convolution over the fast time scale, then we must be more careful because the ϵv_t term is important. Because the update has to be relatively large,² we have to use the u -field to cancel the effect in the u -residual of updates from the v equation.

Here if the convolution is the first fast-time convolution, $p=\{\}$, then choose the correct scale in `small` as then the convolution is only $\mathcal{O}(\sqrt{\epsilon})$.

```

>> linear noise effects <<+
, vvinv(sin(~m*x)*phi(~n,~p),xt)
  => phi(n,(beta(m)/eps).p)*sin(m*x)/eps
  /(if p={} then small else 1)
  when p={} or deg(1/first(p),eps)=1
, uvinv(sin(~m*x)*phi(~n,~p),xt)
  => phi(n,(beta(m)/eps).p)*sin(m*x)/beta(m)
  *(if p={} then small else small^2)

```

²I conjecture that it is this that affects the management of the smallness parameter.

```
when p={} or deg(1/first(p),eps)=1
```

The critical mode is no different when forcing in the v-residual.

```
>> linear noise effects <<+
, vvinv(sin(x)*phi(~n,~p),xt)
  => phi(n,(beta(1)/eps).p)*sin(x)/eps
  /(if p={} then small else 1)
  when p={} or deg(1/first(p),eps)=1
, uvinv(sin(x)*phi(~n,~p),xt)
  => phi(n,(beta(1)/eps).p)*sin(x)/beta(1)
  *(if p={} then small else small^2)
  when p={} or deg(1/first(p),eps)=1
};
```

Second deal with a variety of quadratic noise terms. When a noise term in the residual is a convolution over the slow-scale, then the convolution is smooth and its time derivative correspondingly of the same order so that the ϵv_t causes no problem.

```
>> quadratic noise effects <<+
let { vvinv(sin(~m*x)*zz(~n,~p),xt)
  => zz(n,p)*sin(m*x)/beta(m) when deg(1/first(p),eps)=0
, uvinv(sin(~m*x)*zz(~n,~p),xt) => 0 when deg(1/first(p),eps)=0
, vvinv(sin(~m*x)*phi(~n,~p)^2,xt)
  => phi(n,p)^2*sin(m*x)/beta(m)
  when p neq{} and deg(1/first(p),eps)=0
, uvinv(sin(~m*x)*phi(~n,~p)^2,xt) => 0
  when p neq{} and deg(1/first(p),eps)=0
, vvinv(sin(~m*x)*phi(~n,~p)*phi(~l,~q),xt)
  => phi(n,p)*phi(l,q)*sin(m*x)/beta(m)
  when p neq{} and q neq{}
  and deg(1/first(p),eps)+deg(1/first(q),eps)=0
, uvinv(sin(~m*x)*phi(~n,~p)*phi(~l,~q),xt) => 0
```

```

when p neq{} and q neq{}
and deg(1/first(p),eps)+deg(1/first(q),eps)=0

```

The critical mode is no different when forcing in the v-residual.

```

    ▷▷ quadratic noise effects ◁◁+
, vvinv(sin(x)*zz(~n,~p),xt)
  => zz(n,p)*sin(x)/beta(1) when deg(1/first(p),eps)=0
, uvinv(sin(x)*zz(~n,~p),xt) => 0 when deg(1/first(p),eps)=0
, vvinv(sin(x)*phi(~n,~p)^2,xt)
  => phi(n,p)^2*sin(x)/beta(1)
  when p neq{} and deg(1/first(p),eps)=0
, uvinv(sin(x)*phi(~n,~p)^2,xt) => 0
  when p neq{} and deg(1/first(p),eps)=0
, vvinv(sin(x)*phi(~n,~p)*phi(~l,~q),xt)
  => phi(n,p)*phi(l,q)*sin(x)/beta(1)
  when p neq{} and q neq{}
  and deg(1/first(p),eps)+deg(1/first(q),eps)=0
, uvinv(sin(x)*phi(~n,~p)*phi(~l,~q),xt) => 0
  when p neq{} and q neq{}
  and deg(1/first(p),eps)+deg(1/first(q),eps)=0
};

```

Critical: linearise convolutions over different time scales I contend that we also want to simplify the convolutions because convolutions of the fast time scale ϵ are qualitatively different from convolutions over slow time scales. Thus we do the following ‘linearisation’ of convolutions: whenever the first two convolutions are over different time scales, we transform the convolution into the sum of two convolutions. Change of variables in integration shows that

$$e^{-\alpha t} \star e^{-\beta t} \star = \frac{1}{\beta - \alpha} \left[e^{-\alpha t} \star - e^{-\beta t} \star \right].$$

I have not used this transform in other applications because of the necessity to avoid division by zero when $\alpha = \beta$.³ Here, we are concerned with convolutions over different time scales and so use this formula where, for example, rate β is replaced by fast rate β/ϵ :

$$\begin{aligned} e^{-\alpha t} \star e^{-\frac{\beta}{\epsilon} t} &= \frac{1}{\frac{\beta}{\epsilon} - \alpha} \left[e^{-\alpha t} \star e^{-\frac{\beta}{\epsilon} t} \right] \\ &= \frac{\epsilon/\beta}{1 - \epsilon\alpha/\beta} \left[e^{-\alpha t} \star e^{-\frac{\beta}{\epsilon} t} \right]. \end{aligned}$$

Thus we need to divide by $1 - r$ for various $r = \epsilon\alpha/\beta$ so store its power series in the variable `geom`, with `small` to account for the powers of ϵ . *The correctness of the following transformation is critical.*

```

>> linear noise effects <<+
geom:=for n:=0:deg((1+small^2)^9,small)/2 sum (rat*small^2)^n$
let { phi(~n,~p) => (phi(n,first(p).rest(rest(p)))
    -phi(n,second(p).rest(rest(p)))*(if rest(rest(p))={}
        or deg(1/first(rest(rest(p))),eps)=1
        then small else small^2)
    )*sub(rat=first(p)/second(p),geom)/second(p)
when length(p)>1 and deg(1/first(p),eps)=0
    and deg(1/second(p),eps)=1
, phi(~n,~p) => (phi(n,second(p).rest(rest(p)))
    -phi(n,first(p).rest(rest(p)))*(if rest(rest(p))={}
        or deg(1/first(rest(rest(p))),eps)=1
        then small else small^2)
    )*sub(rat=second(p)/first(p),geom)/first(p)
when length(p)>1 and deg(1/first(p),eps)=1
    and deg(1/second(p),eps)=0
};

```

³However, in general, maybe I should do this linearisation in order to reduce expressions to a more canonical form.

2.2 Update from residuals of the fast equation

The parameter `small`, controls the truncation in nonlinearity and in small parameters. The iteration terminates when the residual of the reaction diffusion equation is zero to the specified order of nonlinearity. For some reason we need to do something nontrivial to the residual in order to force cancellation of terms so I apply `trigsimp`. Note the multiplication and division by `small` in order to cater for other divisions by `small` affecting the error truncation.

```

    >> update from fast residual <<
    resv:=-small^2*eps*df(v,t)+df(v,x,2)-v+u
        +small*rootepts*sig*noise;
    resv:=trigsimp(small^2*resv,combine)/small^2;
    write lengthresv:=length(resv);
    u:=u+uvinv(resv,xt);
    v:=v+vvinv(resv,xt);

```

2.3 Update from residuals of the slow equation

Similarly update from the residual of the slow equation. Divide the evolution `g` by `small` to best keep track of the correct counting of the ‘order’ of a term.

```

    >> update from slow residual <<
    sinu:=trigsimp(u-u^3/6+u^5/120-u^7/5040,combine);
    resu:=-df(u,t)+df(u,x,2)+lamb*sinu-v;
    resu:=trigsimp(small^2*resu,combine)/small^2;
    write lengthresu:=length(resu);
    g:=g+(gd:=secular(small^2*resu,xt)/small^2)/small;
    u:=u+uuinv(resu-gd*sin(x),xt);
    v:=v+vuinv(resu-gd*sin(x),xt);

```


Executing the resultant code constructs the superslow model of the stochastic bifurcation in the fast-slow system of SPDEs.

Acknowledgement This research is supported by Australian Research Council grants DP0774311 and DP0988738.

References

- [1] Xu Chao and A. J. Roberts. On the low-dimensional modelling of Stratonovich stochastic differential equations. *Physica A*, 225:62–80, 1996. [http://dx.doi.org/10.1016/0378-4371\(95\)00387-8](http://dx.doi.org/10.1016/0378-4371(95)00387-8).
- [2] A. J. Roberts. Low-dimensional modelling of dynamics via computer algebra. *Computer Phys. Comm.*, 100:215–230, 1997. [http://dx.doi.org/10.1016/S0010-4655\(96\)00162-2](http://dx.doi.org/10.1016/S0010-4655(96)00162-2).
- [3] A. J. Roberts. Computer algebra resolves a multitude of microscale interactions to model stochastic partial differential equations. Technical report, [<http://eprints.usq.edu.au/archive/00001242/>], December 2005.
- [4] A. J. Roberts. Resolving the multitude of microscale interactions accurately models stochastic partial differential equations. *LMS J. Computation and Maths*, 9:193–221, 2006. <http://www.lms.ac.uk/jcm/9/lms2005-032>.
- [5] A. J. Roberts. Normal form transforms separate slow and fast modes in stochastic dynamical systems. *Physica A*, 387:12–38, 2008.
- [6] Wei Wang, A. J. Roberts, and Jinqiao Duan. Large deviations for slow-fast stochastic partial differential equations. Technical report, University of Adelaide, <http://arxiv.org/abs/1001.4826>, 2010.